A Probabilistic Software Quality Model for C# – an Industrial Case Study

Péter Hegedűs

Both for software developers and managers it is crucial to have clues about different aspects of the quality of their systems. The information can mainly be used for making decisions, backing up intuition, estimating future costs and assessing risks. The ISO/IEC 9126 standard [1] defines six high-level product quality characteristics which are widely accepted both by industrial experts and academic researchers. These characteristics are: *functionality*, *reliability*, *usability*, *efficiency*, *maintainability* and *portability*. The characteristics are affected by low-level quality properties that can be *internal* (measured by looking inside the product, e.g. by analyzing the source code) or *external* (measured by execution of the product, e.g. by performing testing).

Maintainability is probably the most attractive, observed and evaluated quality characteristic of all. The importance of maintainability lies in its very obvious and direct connection with the costs of altering the behavior of the software. Although, the quality of source code unquestionably affects maintainability, the standard does not provide a consensual set of source code measures as internal quality properties. The standard also does not specify the way how the aggregation of quality attributes should be performed. These are not deficiencies of the standard, but it offers a kind of freedom to adapt the model to specific needs.

We have introduced a practical quality model in one of our previous works [2] that differs from the other models in many ways:

- It uses a large number of other systems as benchmark for the qualification.
- The approach takes into account different opinions of many experts, and the algorithm integrates the ambiguity originating from different points of view in a natural way.
- The method uses probabilistic distributions instead of average metric values, therefore providing a more meaningful result, not just a single number.

Although the introduced model proved to be useful and accepted by the scientific community, real industrial settings and evaluations are required to show that our solution is useful and applicable in real environments too. Additionally, the first published model is only a prototype for the Java language and weighted by a small number of researchers and practitioners. The goal of my presented work is to develop a method and model for estimating the maintainability of the C# systems of a large international company. To achieve this goal, the following tasks were completed:

- Together with the industrial partners, we have introduced a new maintainability model for systems written in the C# language.
- A benchmark from the C# systems of the company has been created (several millions of C# code lines has been analyzed).
- A method and tool has been developed for qualifying the smaller components of the company's software using the benchmark - producing a relative measure for maintainability of the components (we were able to rank the components of the company).
- A new weighting has been created involving the developers and managers.
- According to the method and model, a large number of components have been evaluated.

The results were discussed after the evaluation and compared to the developers' opinions. The industrial application of our method and model was successful as the opinions of the developers highly correlated with the maintainability values produced by our C# maintainability model. This result shows that our probabilistic quality model is applicable in the industry, as the industrial partners accepted the provided results and found our approach and tool very useful.

Acknowledgements

This research was supported by the Hungarian national grants GOP-1.1.1-11-2011-0038, GOP-1.1.1-11-2011-0006, and OTKA K-73688.

References

[1] ISO/IEC, ISO/IEC 9126. Software Engineering - Product quality. *ISO/IEC*, 2001.

[2] T. Bakota, P. Hegedűs, P. Körtvélyesi, R. Ferenc, and T. Gyimóthy. A probabilistic software quality model, *in proceedings of the 27th IEEE International Conference on Software Maintenance (ICSM)*, sept. 2011, p. 243-252.